

Docket No. 60000-0012

Patent

UNITED STATES PROVISIONAL PATENT APPLICATION

FOR

APPROACH FOR GENERATING RULES

INVENTORS:

CARLOS A. SERRANO-MORALES

DAVID M. MELLOR

CHRIS W. WERNER

JEAN-LUC M. MARCE

MARC LERMAN

PREPARED BY:

HICKMAN PALERMO TRUONG & BECKER, LLP

1600 WILLOW STREET

SAN JOSE, CALIFORNIA 95125

(408) 414-1080

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EL652871905US

Date of Deposit MAY 9, 2001

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

TIRENA SAY

(Typed or printed name of person mailing paper or fee)

Tirena Say

(Signature of person mailing paper or fee)

60000-0012-050901

APPROACH FOR GENERATING RULES

RELATED APPLICATIONS

This application claims priority of U.S. Provisional Application No. 60/203,073,
5 entitled, "Flexible And Adaptable Rule Architecture And Development Environment", filed
by Carlos Serrano-Morales, et al. on May 9, 2000, the contents of which are hereby
incorporated by reference.

FIELD OF THE INVENTION

The invention is generally related to information management systems and more
10 particularly, to a computer implemented approach for generating business rules.

BACKGROUND OF THE INVENTION

Many decisions made by business enterprises can be represented by one or more
business rules. As used herein the term "rule" refers to one or more actions or operations that
are performed upon the satisfaction of one or more conditions. Thus, a "business rule" refers
15 to one or more business actions or business operations that are performed upon the
satisfaction of one or more conditions. For example, in the context of a financial company
that issues credit cards, the process for determining whether to offer a credit card to a
particular individual based upon particular financial or demographic information can be
represented by one or more business rules.

20 Business rules are important because they allow business decisions to be automated
using computer software. The business logic and conditions defined by business rules are
embodied in computer software. Referring to the prior example, it is not uncommon for
financial companies to automate, using computer software, the decision about whether a

credit card should be offered to a particular individual. Conventionally, these types of computer systems use complex customized software to implement business rules.

A significant problem with using custom software to implement business rules is that the people in business organizations who decide on changes to business rules generally cannot themselves implement those changes in the computer software. When business logic or conditions change, the computer software must be updated to reflect the change. For example, in the context of issuing credit cards, if the minimum salary requirement is changed from X to Y, then the software must be manually updated to reflect the change in the minimum salary requirement from X to Y. Updating computer software generally requires technical expertise that the business people who decide the business rules simply don't have. These people are often financial analysts or high-level managers. Updating the software typically involves changing values in source code or data files and then "rebuilding" the software, which requires recompiling source code to generate object code and then linking the object code with libraries of other object code to generate a new runtime executable.

Some companies use a database system to store condition values so that the condition values can be updated and used by application software without having to rebuild the application software. When condition values are updated, the software retrieves the updated values from the database. This solves the problem of having to rebuild application software when condition values change, but still does not allow business people who define business rules and changes to implement those changes themselves. Furthermore, databases are generally not capable of storing the complex business logic required by business rules. Rather, the business logic must still be embodied in the complex custom software. Hence, changing business logic rules requires updating the custom software, which cannot be done by non-technical personnel.

Based upon the need for non-technical people to be able to update business rules and applications that use the business rules, an approach for updating business rules and applications that use them that does not suffer from limitations of prior approaches is highly desirable.

SUMMARY OF THE INVENTION

Techniques are provided for facilitating the generation and updating of rules, allowing non-technical users to create and modify business rules. According to an aspect of the invention, rule element data is generated that describes a set of choices for a rule element of a rule. User interfaces may be generated based on the rule element data. The user

5 interfaces are used to collect user choice data that specifies choices for the rule element chosen by the user.

According to another aspect of the invention, the rule element data may define the set of choices as a set of constant values, or use data from a database system to define the choices. The rule element data may also define a set of conditions (e.g. logical expressions)

10 that may be chosen as a rule element by a user.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

5 FIG. 1A is a block diagram depicting a system for generating rules that are stored in a rules repository according to an embodiment of the invention;

 FIG. 1B is a diagram depicting a rule according to an embodiment of the invention;

 FIG. 2 is a block diagram depicting a rule template according to an embodiment of the invention;

10 FIG. 3 is a block diagram depicting rule element providers that describe choices for editable rule elements according to an embodiment of the invention;

 FIG. 4 is a block diagram depicting rule element holders used to describe editable rule elements according to an embodiment of the invention;

15 FIG. 5 is a block diagram depicting the screen of user interfaces for collecting user choices for an editable rule element according to an embodiment of the invention;

 FIG. 6 is a block diagram depicting a process for generating rules according to an embodiment of the invention;

 FIG. 7 is a block diagram depicting templates used to define selectable conditions for an editable rule element according to an embodiment of the invention; and

20 FIG. 8 is a block diagram depicting a computer system upon which embodiments of the invention may be implemented.

DETAILED DESCRIPTION OF THE INVENTION

A method and apparatus for generating and updating rules is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the invention. It will be apparent, however, that the invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the invention. Various aspects of the invention are described hereinafter in the following sections: 1) overview; 2) rule elements; 3) illustrative template; 4) rule element provider types and structure; 5) rule element holders; 6) illustrative user interface; 7) creating rules from templates; 8) conditions as editable rule elements; 9) advantages; and 10) implementation mechanisms.

1. Overview

A novel approach is described for generating and updating rules that allows non-technical users to create and modify rules. FIG. 1A is a block diagram depicting an example architecture of a rules generation system 100, which is a system that facilitates generating and updating rules according to an embodiment of the invention. Rules generation system 100 includes various components: a rules engineer interface 102, user interfaces 104, and a rules generator 106. These components generate various entities which are stored in repository 108. Repository 108 may be any storage mechanism, including database systems and repositories that store data according to the XML standard.

Rules engineer interface 102 is an interface through which a rules engineer, such as rules engineer 110, may generate templates and providers 112, which are stored in repository 108. Templates and providers 112 contain class templates 114, rule templates 116, ruleflow

templates 118, other types templates not shown (e.g. function templates), and rule element providers 120. In general, a template is data that describes an entity that may be executed by a rules engine. Class templates 114, for example, define classes.

Rule templates 116 define rules. In particular, a rule template specifies a rule structure, rule elements that may be chosen by a user 122, and rule elements that may not be chosen by user 122. Rule elements are the elements that form a rule. Examples of rule elements include, without limitation, operators, variables, constants, conditions, and actions. A template may define a subset of rule elements that comprise a complete structure for a rule.

Ruleflow templates 118 is a set of ruleflow templates. A ruleflow template is data that defines a ruleflow. A ruleflow is a flow of execution of a set of tasks, where the set of tasks includes a task involving the execution of a group of rules. According to an embodiment of the present invention, rules tasks entail the execution of rules associated with a group of templates.

Rule element providers 120 define a set of choices that a user may choose for a particular editable rule element defined by a template. For example, a rule element provider may define a set of strings that a user 122 may choose for a rule element.

User interface generator 104 generates user interfaces 124. User interface generator 104 creates user interfaces 124 to guide a user, such as user 122, to create or edit rules according to definitions specified by rule templates 116 and rule element providers 120. Once generated, user interfaces 124 present editable rule elements that may be edited by the user, and present and limit the choices the user may choose for the rule elements. For example, an editable rule element may be a string value representing the abbreviation of a state. In this context, a user interface may include a list box control that allows a user to choose a value for the editable rule element. Thus, in this example, the list box control displays a list of states that may be selected by the user. In this way, the user interface presents and limits the choices the user may choose

for the editable rule element. User interfaces 124 insulate user 122 from many of the complicated details of engineering rules.

User interfaces 124 generate rule element choice data that represents the choices made by a user for editable rule elements. The rule element choice data is stored as one or more

5 “instances of a template” (herein template instances). Template instances 126 represent a set of choices a user makes for the one or more editable rule elements defined by a rule template of rule templates 116.

According to an embodiment of the invention, user interfaces 124 are embodied as one or more Web pages. These web pages may contain embedded code, scripts, or programs that
10 are executed at a client, such as a browser. These programs can be, for example, Java applets, Java scripts, or Active X controls. The programs may be stored temporarily in the cache of a client, or more permanently as, for example, one or more plug-in applications.

Rules generator 106 generates rules and stores the rules as rules 128 in repository 108. Rules generator 106 generates rules 128 according to rule structures specified by rule templates
15 116 and the choices for editable rule elements specified by template instances 126.

Rules generator 106 may also use other information to generate rules. For example, rules generator 106 may use rule templates 116, template instances 126 and other rule data that specifies one or more attributes of a rule not specified by the template instances 126.

A project, such as project 130, refers to one or more template instances. Project 130
20 refers to template instances 126-1 and 126-2. A project groups template instances into a module from which rules generator 106 generates a set of rules. Rules generator 106 is capable of generating a group of rules from the template instances referred to by a particular project. According to an embodiment of the invention, a set of template instances, such as template instances 126-1, may be stored in a data file. Rules generator 106 may generate a group of

rules from a particular data file.

2. Rule Elements

FIG. 1B is a diagram that depicts rule 150 and its constituent rule elements according to an embodiment of invention. Rule 150 may be written according to a rules language, for example the structured rule language (SRL), defined by Brokat Technologies Incorporated. Rules written in SRL may be transformed, e.g., compiled, into another form, e.g., code that is executable by a rules engine.

Referring to FIG. 1B, rule 150 includes rule elements 152, 154, 156, 158, 160, 162 and 164. Rule element 152 is a variable in the form of an attribute of an object, in particular, the shoppingCart attribute of the object theCustomer. Rule element 154 is an operator, in particular, the contains operator. Rule elements 156 and 158 are constants; rule element 156 is the string constant "cigars", rule element 158 is the constant integer 20. Rule element 160 is a condition in the form of an expression formed by rule elements 152, rule element 154, and rule element 156. Rule element 162 is another example of a condition. Rule element 164 is an action in the form of a method invocation, i.e., an invocation of the method append of object promotions.

Rules, rule elements, and rules engines are discussed in Blaze Advisor™, Technical White Paper, by Blaze Software™, Inc., 1999, the contents of which are hereby incorporated by reference.

3. Illustrative Template

FIG. 2 is a block diagram depicting an illustrative template and rule element providers in greater detail. Referring to FIG. 2, rule template 200 is comprised of three types

of data: rule element holders 202, rule structure 204, and user interface definition 206. User interface definition 206 defines aspects of user interfaces 124.

Rule structure 204 generally defines a rule structure that specifies non-editable rule elements and locations in the rule structure that are occupied by editable rule elements. Rule structure 204 may include SRL code that defines non-editable elements. An editable rule element's position within a rule structure is specified by a placeholder such as placeholders 208, 210, and 212. According to one embodiment of the invention, placeholders are embedded in SRL code.

In the present example, placeholders 208, 210, and 212 refer to rule element holders ShoppingCartItemHolder 214, AgeHolder 216, and EmploymentStatusHolder 218, respectively. A rule element holder may define characteristics about an editable rule element. For example, a rule element holder may define whether an editable rule element is a string value, a collection of string values, an operator, or a condition. A rule element holder may also identify a template that defines a portion of a rule structure, as shall be described in greater detail herein after.

Rule element holder ShoppingCartItemHolder 214, AgeHolder 216, and EmploymentStatusHolder 218 each refer to a rule element provider. A rule element provider defines a set of choices a user may choose for an editable rule element. The rule element provider also constrains the choices a user may choose for the rule element. For example, a rule element provider may define a list of string values (abbreviations of states) as a set of choices for an editable rule element representing a state. The rule element provider thus defines the list as the set of choices for the editable rule element, and constrains the user to members of the list. In this present example, ShoppingCartItemHolder 214 refers to ShoppingCartItemProvider 220, AgeHolder 216 refers to AgeProvider 222,

EmploymentStatusHolder 218 refers to EmploymentStatusProvider 224. AgeProvider 222 restricts the value of an editable rule element to an integer value between 20 and 100.

4. Rule Element Provider Types And Structure

There are various types of rule element providers. The structure of a rule element provider and the information it contains depends on its type. FIG. 3 is a block diagram depicting rule element providers ShoppingCartItemProvider 220, AgeProvider 222, and EmploymentStatusProvider 224 in greater detail to illustrate example types of rule element providers and structures.

ShoppingCartItemProvider 220 is a list rule element provider. A list rule element provider defines a list of data value choices for a rule element. ShoppingCartItemProvider 220 includes a data type 302 that defines the data type of the values in the list, which in this case is string. Value list 304 contains the list of string values. The list of string values, which represent employment status, include 'Self Employed', 'Part Time', and 'Full Time'.

AgeProvider 222 is an integer rule element provider. An integer rule element provider defines a range of value choices for an editable rule element. Beginning 306 defines the beginning of the range and ending 308 defines the end of the range. AgeProvider 222 (FIG. 2) is used to constrain the value of an editable rule element corresponding to a human age. Accordingly, beginning 306 is set to 0 and ending 308 to xxx.

EmploymentStatusProvider 224 is a database provider. A database provider defines a set of data value choices for a particular rule element, where the set is the result set of a database query. Connect details 310 contain information about how to connect to a database. This information includes, for example, the location of the database server, and a user name and password. Query 312 is data defining a query. Such data may be a query string written

in SQL. For example, EmploymentStatusProvider 224 defines a collection of data strings describing products, where the collection is the result set of a query issued against a database table containing product names.

Other types of rule element providers may define a set of data value choices from sources other than database systems. For example, a rule element provider may use the values in a data file or other repository to define the particular data value choices for a editable rule element.

5. Rule Element Holders

FIG. 4 is a block diagram depicting an example implementation of ShoppingCartItemHolder 214. In this example, ShoppingCartItemHolder 214 includes data elements provider-template 400, and number of items 402. Provider-template 400 will identify either a rule element provider or a template, depending on the rule element holder's type. According to an embodiment of the present invention, a rule element holder may have one of several types: a provider type or a template.

ShoppingCartItemHolder 214 is an example of a provider type of rule element holder. A provider type identifies a rule element provider. In this case, the data element provider-template 400 identifies ShoppingCartItemProvider 220. The identified rule element provider defines a set of choices for the editable rule element corresponding to the rule element holder. In addition, because provider-template 400 refers to ShoppingCartItemProvider 220, the type of editable rule element defined by ShoppingCartItemHolder 214 is string. In this way, a rule element holder defines the type of an editable rule element.

A rule element holder may also define an editable rule element as a collection of values. Number of items 402 specifies the number values in the collection. For example,

suppose that number of items 402 is set to 5. In this case, ShoppingCartItemHolder 214 defines an editable rule element that may be a collection of up to five values. If number of items 402 were set to 1, then the editable rule element would not be a collection. In addition, number of items 402 may contain a flag to indicate whether or not the values in the collection should be unique.

For rule element holders of the template type, provider-template 400 refers to another template that defines a portion of a rule structure, which may include non-editable rule elements and editable rule elements. These other templates are useful for defining a partial rule structure frequently found in numerous rule structures.

6. Illustrative User Interface

As mentioned previously, rules generation system 100 generates user interfaces that accept choices for editable rule elements. The user interfaces are generated according to definitions specified by a particular template. FIG. 5 is a block diagram that depicts a screen for a user interface generated for rule template 200.

Referring to FIG. 5, screen 500 includes various graphical controls used to accept rule element choices from a user. Control 502 is a list box control. Label control 504 is displayed in association with control 502. Control 502 includes list 506. List 506 displays the rule element choices defined by ShoppingCartItemProvider 220 for ShoppingCartItemHolder 214. A user may only select a choice in list 506. The choices in the list are those belonging to the result set of query 312.

Control 508 is a text box control used for collecting integer rule elements representing an age. The user interface is configured to only accept the range specified by AgeProvider 222. i.e. 20 through xxx.

Control 510 is a list box control that includes list 514. List 514 displays the rule element choices defined by EmploymentStatusProvider 224 for EmploymentStatusHolder 218. The user may only select from one of the choices in the list, the choices being defined by EmploymentStatusProvider 224.

5 Submit button 516 is a command button manipulated by a user to store the rule element choices made through the interface. In response to manipulating the submit button 516, rules generation system 100 stores the choices as a template instance of rule template 200.

10 User interface definition 206 stores data that controls various characteristics of screen 500. For example, user interface definition 206 may include data specifying the background color of screen 500, as well as the font and font size of the labels.

7. Creating Rules From Templates

FIG. 6 is a flow diagram depicting a process for facilitating the generation and updating of rules using various components described previously, according to one
15 embodiment of the invention.

Referring to FIG. 6, at step 600, rules engineer 110 interacts with the rules engineer interface 102 to generate rule template 200.

At step 602, rules generation system 100 creates a user interface that collects from user 122 rule element choices according to rule template 200.

20 At step 604, user 122 interacts with the user interface to generate rule element choices. Screen 500, as depicted in FIG. 5, shows the choices made by user 122. In response to manipulating submit button 505, the choices are stored as a rule template instance having the values shown in screen 500. Specifically, the editable rule elements that correspond to

the following rule element holders have the following values: (1) for ShoppingCartItemHolder 214, 'Cognac' (2) for AgeHolder 216, '20' and (3) for EmploymentStatusHolder 218, 'FullTime'.

At step 606, rules generation system 100 generates rules based upon rule template 200 and Template instances 126. Rule 150 (FIG. 1B) represents the rule generated for the current example.

8. Conditions As Editable Rule Elements

Techniques for facilitating the generation and updating of rules have been illustrated using editable rule element types that are constant strings and integers. However, the techniques described herein may be used for other types of rule elements, such as operators and conditions. FIG. 7 illustrates a template that defines an editable rule element as a condition.

FIG. 7 shows some components of template 700, which define a rule structure having an editable rule element as a condition. Referring to FIG. 7, placeholder 702 is a placeholder within rule structure 704. Placeholder 702 refers to ConditionHolder 706, which refers to ConditionProvider 708.

ConditionProvider 708 defines a set of conditions which may be selected for an editable rule element. Specifically, ConditionProvider 708 includes template list 710, which contains template references 710-1 through 710-N. Each of template references 710-1 through 710-N refers to a condition template in condition templates 712. A condition template defines a rule structure for a condition, which may specify an expression or other partial rule structure. A condition template may also define editable rule elements.

A user interface for template 700 may include multiple screens, such as a screen for template 700 and a screen for each of the condition templates. When a user chooses a particular condition defined by condition template 700, a screen generated for that condition template is presented to the user. The screen may include graphic controls for editable rules elements defined by the condition template.

9. Advantages

The approaches described for generating and updating rules have various advantages. Specifically, the approach allows technical and non-technical users to participate in the process of generating and updating rules without the need for skills and knowledge outside their respective bailiwicks. A non-technical business analyst may provide values and conditions for business rules through user interfaces configured to lead them through this process. Providing the values and conditions for rule elements thus requires no programming of business rules. The technical person who develops business rules does not need to know specific values or conditions needed for business rules.

Moreover, the degree of participation between technical and non-technical persons is reduced. For example, implementing changes to rules may require updating new rules with new values determined by a business analyst. These values do not have to be communicated to technical persons to reprogram business rules. Rather, the business analyst merely enters the values through user interfaces. A process that requires less participation between individuals is easier to manage because the process requires less coordination between them.

The approach may require less work on the part of technical persons to develop business rules. The technical person does not have to program each rule, only its architecture.

Even the programming needed to develop user interfaces is reduced because the code for the user interfaces are generated automatically based on the templates.

Finally, external sources, such as database systems, may serve as sources for choices and constraints for editable rule elements. Modifications to data in the database systems affecting those choices are automatically reflected in the user interfaces through which the choices are made, without requiring any reprogramming on the part of technical persons or further data entry on the part of business analysts.

10. Implementation Mechanisms

The approach described herein for generating and editing rules is applicable to a wide variety of contexts and implementations depending upon the requirements of a particular application and the approach is not limited to any particular contexts and implementations. For example, the approach may be integrated into a rules architecture environment or a rules engine. As another example, the approach may be implemented as a stand-alone mechanism that is communicatively coupled to a rules architecture environment or rules engine. The approach may be implemented at a single location or may be implemented in a distributed computing environment. The approach may be implemented in computer hardware, software or firmware, or in any combination of computer hardware, software and firmware.

FIG. 8 is a block diagram that illustrates a computer system 800 upon which an embodiment of the invention may be implemented. Computer system 800 includes a bus 802 or other communication mechanism for communicating information, and a processor 804 coupled with bus 802 for processing information. Computer system 800 also includes a main memory 806, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 802 for storing information and instructions to be executed by processor 804.

Main memory 806 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 804. Computer system 800 further includes a read only memory (ROM) 808 or other static storage device coupled to bus 802 for storing static information and instructions for processor 804. A storage
5 device 810, such as a magnetic disk or optical disk, is provided and coupled to bus 802 for storing information and instructions.

Computer system 800 may be coupled via bus 802 to a display 812, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 814, including alphanumeric and other keys, is coupled to bus 802 for communicating information and
10 command selections to processor 804. Another type of user input device is cursor control 816, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 804 and for controlling cursor movement on display 812. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

15 The invention is related to the use of computer system 800 for implementing the techniques described herein. According to one embodiment of the invention, those techniques are performed by computer system 800 in response to processor 804 executing one or more sequences of one or more instructions contained in main memory 806. Such instructions may be read into main memory 806 from another computer-readable medium,
20 such as storage device 810. Execution of the sequences of instructions contained in main memory 806 causes processor 804 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 804 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 810. Volatile media includes dynamic memory, such as main memory 806.

Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 802. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 804 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 800 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 802. Bus 802 carries the data to main memory 806, from which processor 804 retrieves and executes the instructions. The instructions received by main memory 806 may optionally be stored on storage device 810 either before or after execution by processor 804.

Computer system 800 also includes a communication interface 818 coupled to bus 802. Communication interface 818 provides a two-way data communication coupling to a network link 820 that is connected to a local network 822. For example, communication interface 818 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 818 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 818 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 820 typically provides data communication through one or more networks to other data devices. For example, network link 820 may provide a connection through local network 822 to a host computer 824 or to data equipment operated by an Internet Service Provider (ISP) 826. ISP 826 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 828. Local network 822 and Internet 828 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 820 and through communication interface 818, which carry the digital data to and from computer system 800, are exemplary forms of carrier waves transporting the information.

Computer system 800 can send messages and receive data, including program code, through the network(s), network link 820 and communication interface 818. In the Internet example, a server 830 might transmit a requested code for an application program through Internet 828, ISP 826, local network 822 and communication interface 818.

The received code may be executed by processor 804 as it is received, and/or stored in storage device 810, or other non-volatile storage for later execution. In this manner, computer system 800 may obtain application code in the form of a carrier wave.

In the foregoing specification, the invention has been described with reference to
5 specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.
